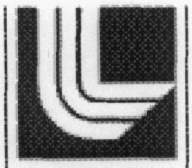


LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-JC-152416

Towards Perceptual Interface for Visualization Navigation with Bezier Curves and Registered 3-D Data

M. C. Shin, L. V. Tsap, D. B. Goldgof

March 20, 2003

IEEE Workshop on Computer Vision and Pattern Recognition for Human Computer Interaction, Madison, WI, June 17, 2003

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Towards Perceptual Interface for Visualization Navigation of Large Data Sets Using Gesture Recognition with Bezier Curves and Registered 3-D Data

Min C. Shin

Department of Computer Science
University of North Carolina at Charlotte
9201 University City Blvd
Charlotte, NC 28223
mcshin@uncc.edu

Leonid V. Tsap

Electronics Engineering Department
University of California Lawrence Livermore National Laboratory
Livermore, CA 94551
tsap@llnl.gov

Dmitry B. Goldgof

Department of Computer Science
University of South Florida
Tampa, FL 34642
goldgof@csee.usf.edu

Abstract[1]

This paper presents a gesture recognition system for visualization navigation. Scientists are interested in developing interactive settings for exploring large data sets in an intuitive environment. The input consists of registered 3-D data. A geometric method using Bezier curves is used for the trajectory analysis and classification of gestures. The hand gesture speed is incorporated into the algorithm to enable correct recognition from trajectories with variations in hand speed. The method is robust and reliable: correct hand identification rate is 99.9% (from 1641 frames), modes of hand movements are correct 95.6% of the time, recognition rate (given the right mode) is 97.9%. An application to gesture-controlled visualization of 3D bioinformatics data is also presented.

EDICS: Primary: 2-MOTD Motion Detection and Estimation, Secondary: 2-SEQP Image Sequence Processing.

[1]This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract number W-7405-Eng-48. UCRL-JC-XXXXXX

Contents

1 Introduction

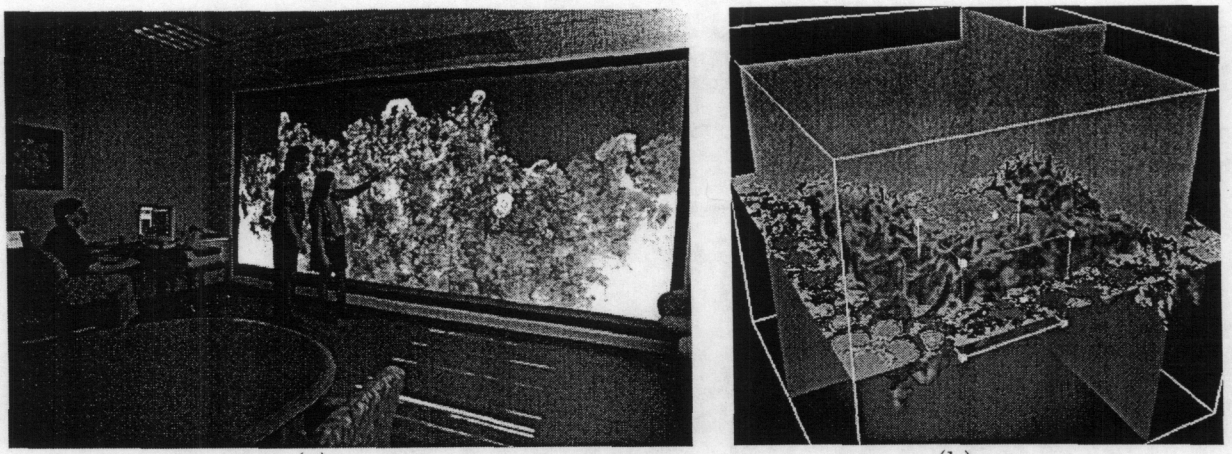
1.1 Motivation

Large and complex data sets are produced at a more rapid pace than tools and algorithms for their processing, analysis, and exploration. For example, the National Institute of Health's (NIH) Visible Human project generated data sets of a single 3-D volume that consists of 12 billion elements. Nearly a terabyte of satellite data is produced on a daily basis. Advanced physics simulation at Lawrence Livermore National Laboratory (LLNL) is generating large data sets, which are expected to increase to one terabyte every five minutes by 2004.

Among the tools used to help explore and understand large data, visualization aids in gaining greater insight into important physical parameters (such as temperature, height, stress, velocity or pressure) and in finding anomalies. Such anomalies often are not obvious to scientists during the automatic localization but are easily picked out with visual data exploration. Correct representation can drastically reduce the time needed for the analysis. As data becomes huge, it requires more time for processing. Even the latest supercomputers may require days or even weeks for computations. This makes real-time visualization mission-critical, since interesting properties can show up which allow scientists to adjust parameters of computation and restart it, if needed. Visualization is integrated into the process and is no longer just the last step.

State-of-the-art visualization displays keep pace with the data requirements. For instance, one of LLNL's "power walls" (Figure 1(a)) is a 15-projector system that displays approximately 19.7 million pixels on a 16- by 8-ft screen. Systems like this allow for detailed data analysis and team collaboration. However, applying even simple commands to the data (such as zoom, rotation and translation) requires a secondary, or "background," communication process between scientists working with the data (see the two standing by the screen in Figure 1(a)) and the "operator" responsible for executing selected commands (sitting, left). This reduces productivity of the team and affects quality of presentations.

Therefore, scientists are interested in developing new, interactive settings for exploring their data in a more intuitive environment. A gesture-recognition system can interpret commands and supply data manipulation parameters to visualization software (Figure 1(b)) without having an "operator" involved in the process.



(a) (b)

Figure 1: (a) A 15-projector display with a total resolution of 6,400 by 3,072 used for interactive applications. (b) An example of interactive manipulation of isosurface and volume rendering parameters.

Since the system is being developed as a front end for gesture-controlled, large-scale visualization and virtual reality manipulation, certain requirements and complications are apparent. First, 3-D information is required, not necessarily at a video-frame rate, but at least a few times per second (optimal parameters should be determined as a result of testing on a large group of people). Second, traditional techniques such as background subtraction cannot easily separate a figure from the background, since the entire body of the interacting person (not only arms or hands) is moving. Moreover, interaction takes place in front of the screen where the data is updated dynamically and, therefore, the background changes most of the time. Third, motion of the interacting person should be natural and should result in intuitive data manipulation, where intuitive means easily learned and fast to provide immediate results.

1.2 Previous Work

Gesture tracking and recognition are important research domains. Traditional approaches to tracking typically relied on segmentation of the intensity data, using motion or appearance data. A majority of the methods began by segmenting the human body from the background. For example, in "blob approaches," people were modeled as a number of blobs resulting from pixel classification based on their color and position in the image. Wren *et al.* [1] achieved segmentation by classifying pixels into one of several models, including a static world and a dynamic user represented by gaussian blobs. Yang and Ahuja [2] used skin color and the geometry of palm and face regions for segmentation stages of their system. A Gaussian mixture (with parameters estimated by an EM algorithm) modeled the distribution of skin-color

pixels. Rehg and Kanade [3] used a 3-D hand model to track a hand. They compared line features from the images with the projected model and performed incremental state corrections. Similar work was presented by Kuch and Huang [4], in which the synthesis process could fit the hand model to any person's hand. Cutler and Davis [5] segmented the motion and computed a moving object's self-similarity (including human motion experiments).

A significant amount of work is being performed in the area of recognition, where Hidden Markov models (HMMs) are often employed successfully [6, 7, 8] by allowing researchers to address the highly stochastic nature of human gestures. Yacoob and Black proposed parameterized representation of human movement in the form of principal components [9]. Bobick and Wilson [10] treated gesture as a sequence of states and computed configuration states along prototype gestures. Yang and Ahuja [2] used motion trajectories for recognition. Grzeszcuk *et al.* [11] described classification algorithm with statistical moments of the binarized gesture templates. Hong *et al.* [12] treated each gesture as a finite state machine (FSM) in the spatial-temporal space; FSMs were trained using k-means clustering. A preliminary trained neural network was used by Sato *et al.* [13]. Hongo *et al.* [14] performed recognition by a linear discriminant analysis in each discriminant space by using four directional features. Approach described by Yoon *et al.* [15] derived features from location, angle, and velocity and employed a k-means clustering algorithm for the HMMs. Gesture contour representation and alignment-based classification were proposed by Gupta and Ma [16]. A review by Aggarwal and Cai [17] classified approaches to human motion analysis, the tasks involved, and major areas related to human motion interpretation. A review by Pavlovic *et al.* [18] addressed main components and directions in gesture recognition research for human-computer interaction (HCI).

2 Framework

2.1 Overview

In this section, we describe the method for recognizing three gesture types: rotation, zoom, and translation. Given the 3-D trajectory of the manipulating hand, we fit a Bezier curve to the trajectory. The curvature of the curve is used to determine the gesture.

Gesture recognition involves five steps:

1. Detecting the manipulating hand

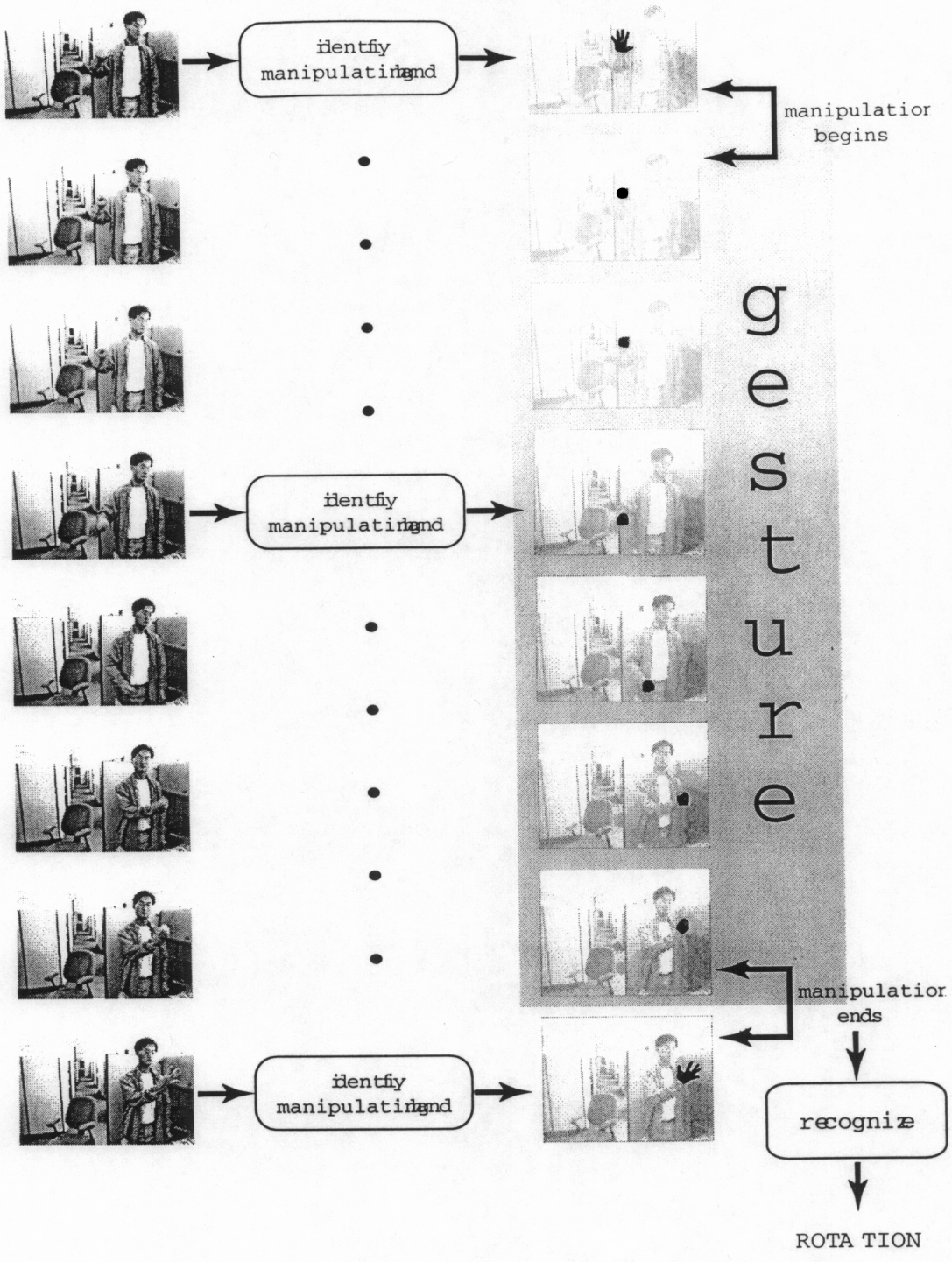


Figure 2: Overview of gesture recognition.

2. Identifying the beginning of the gesture,

3. Detecting the end of the gesture

4. Computing the 3-D trajectory of the manipulating hand

5. Recognizing the gesture by fitting the curve to the 3-D trajectory.

These steps are described in separate sections of this paper. Please see Figure 2 for an overview of the recognition process.

2.2 Manipulating Hand Detection

This presents a brief overview of the process. For more details, please see [20].

Detecting skin regions. To identify a hand involved in virtual object manipulation, the color image is converted to the SCT color space to reduce lighting artifacts [19]. Skin pixels are selected with a minimum distance classifier using Mahalanobis distance (skin statistics is collected off-line). After the noise removal (by a sequence of erosions and dilations), the skin image is segmented using connected component analysis. Small regions or unlikely shapes are removed from further consideration.

Identifying the hand involved in manipulation. We assume that the hand gesture occurs in front of the body. Therefore, the skin region closest to the camera is identified as the manipulating hand. We determine the 3-D location of the region by using histogram-based noise filtering, where all smaller bins are considered noise [20]. Virtual commands do not happen all the time. Hand gestures can be classified in general as object manipulations and other hand movements. Selection of meaningful gestures is described next.

2.3 Gesture On, Gesture Off

We recognize the gestures for manipulation of virtual objects by "grabbing" them. Both grabbing and releasing are detected by checking the area change of the hand. If the area has decreased, then the hand is closed and therefore the object is grabbed. If the area has increased, then the hand is opened and therefore the object is released. Since the speed of the hand closing and opening could vary, we build a history of hand area change. We accumulate the area change on the past three frames. Grabbing is detected when the accumulated change of area is less than -75%, and releasing is detected when the accumulated change is greater than 75%. Note that the negative area change means the area decreased.

Changes due to small movement or noise around the hand area could occur, and these should not be counted toward the determination of grabbing and releasing. Therefore, we ignore change of area smaller than 15%.

2.4 Bezier Curves

Representation and Properties

A Bezier curve uses control points to represent a curve.

$$C(u) = \sum_{i=0}^{n} B_{i,n}(u) P_i \tag{1}$$

Points along a curve are determined by using u, $0 \leq u \leq 1$. n indicates the degree of curve which is one less than the number of control points. P_i is the ith control point where $P(0) = C(0)$ and $P(n) = C(1)$. $B_{i,n}$ is a Bernstein polynomial.

$$B_{i,n} = \frac{n!}{i!(n-i)!} n^i (1-u)^{n-i} \tag{2}$$

We use Bezier curves with three control points describing quadratic curves. The Bernstein polynomials with $n = 2$ are shown below.

$$B_{0,2} = (1-u)^2 \tag{3}$$

$$B_{1,2} = 2u(1-u) \tag{4}$$

$$B_{2,2} = u^2 \tag{5}$$

Therefore, the equation for the Bezier curve with three control points is simplified.

$$C(u) = (1-u)^2 + 2u(1-u) + u^2 \tag{6}$$

Curve Fitting - General Method

We use the curve approximation method described by Piegl and Tiller [21]. We describe the method for fitting curves with n+1 control points with m+1 data points. The simple solution for three control points is shown in Appendix A.

Given a set of 3-D points to fit $Q_0, \ldots, Q_m$, we estimate control points $P_0, \ldots, P_n$ where $m > n$. $\bar{u}_k$ is computed using the length of the curve up to Q_k.

$$|Q| = \sum_{i=1}^{m} |Q_i - Q_{i-1}| \tag{7}$$

$$\bar{u}_k = \frac{\sum_{i=1}^{k} |Q_i - Q_{i-1}|}{|Q|} \tag{8}$$

We first set $P_0 = C(0) = Q_0$ and $P_n = C(1) = Q_m$. We then estimate the unknowns which are the intermediate control points $P_1, \cdots, P_{n-1}$ such that E is minimized.

$$E = \sum_{k=1}^{m-1} |Q_k - C(\bar{u}_k)|^2 \tag{9}$$

For the system with $(n+1)$ control points, the estimation of P becomes solving

$$(N^T N)P = R \tag{10}$$

$$N = \begin{bmatrix} N_{1,p}(\bar{u}_1) & \cdots & N_{n-1,p}(\bar{u}_1) \\ \vdots & \ddots & \vdots \\ N_{1,p}(\bar{u}_{m-1}) & \cdots & N_{n-1,p}(\bar{u}_{m-1}) \end{bmatrix} \tag{11}$$

where R is the vector of $n-1$ points

$$R = \begin{bmatrix} N_{1,p}(\bar{u}_1)R_1 & + \cdots + & N_{1,p}(\bar{u}_{m-1})R_{m-1} \\ & \vdots & \\ N_{n-1,p}(\bar{u}_1)R_1 & + \cdots + & N_{n-1,p}(\bar{u}_{m-1})R_{m-1} \end{bmatrix} \tag{12}$$

Each R_k is defined as

$$R_k = Q_k - N_{0,p}(\bar{u}_k)Q_0 - N_{n,p}(\bar{u}_k)Q_m \tag{13}$$

$$P = \begin{bmatrix} P_1 \\ \vdots \\ P_{n-1} \end{bmatrix} \tag{14}$$

P is a matrix with unknown control points in each row where x, y, and z are stored as a separate element.

2.5 Recognition

Good Input? The gesture is recognized using the trajectory between grabbing and releasing. However, the gesture is ambiguous if the movement is too small. Also, if the trajectory contains abruptly sampled or too few points, computing the shape of trajectory becomes difficult.

We consider the gesture to be invalid if the trajectory is shorter than 20cm, if the trajectory is irregularly sampled, or if the trajectory contains less than six points. We selected the minimum number of points to be six because (1) we observed that at least six points were needed to robustly fit to three control points, and (2) we assumed that the gesture roughly requires one or two seconds and the frame rate was close to six frames/sec.

In order to determine irregularity, we have computed U_k for all the points along the trajectory, then counted the points in three bins ranging from [0.0, 0.33], [0.34, 0.66], and [0.67, 1.0] with U_k values. If

there are no items in any of the three bins, we consider the trajectory to be invalid. However, a natural variability in gesture velocity is allowed. u_k can adjust for different speeds/intervals between points Q_k. u is the sampling parameter which could range between [0, 1] to select a point on a curve by using $C(u)$. $C(0)$ is the beginning of the curve, and $C(1)$ is the end of the curve. Choosing u_k (which is the u for the kth point) by using the distance between two consecutive points is equivalent to adjusting the different speeds along the gesture.

Figure 3: Selected frames for three sample gestures: zoom (top row), translate (mid row), and rotate (bottom row). Hands used for manipulation are (automatically) detected and shown in natural skin colors, whereas the rest of each image is displayed as greyscale for visualization.

Determining Gesture Types. A geometric (rather than statistical) model is motivated by the nature of the application (representation of visualization operations) and its requirements (robustness and simplicity). In order to allow for the variation of the hand speed within a gesture and between gestures, the $\bar{u}_k$ in a Bezier curve description is computed using the distance that the hand travels between the frames.

Using its angle and direction, the trajectory is classified into rotation, translation or zoom. Examples of three gestures are shown in Figure 3. The inside angle at the middle point is used to recognize the gesture from 3-D trajectory (see Figure 4). Using two end control points (P_0, P_2) and the middle point

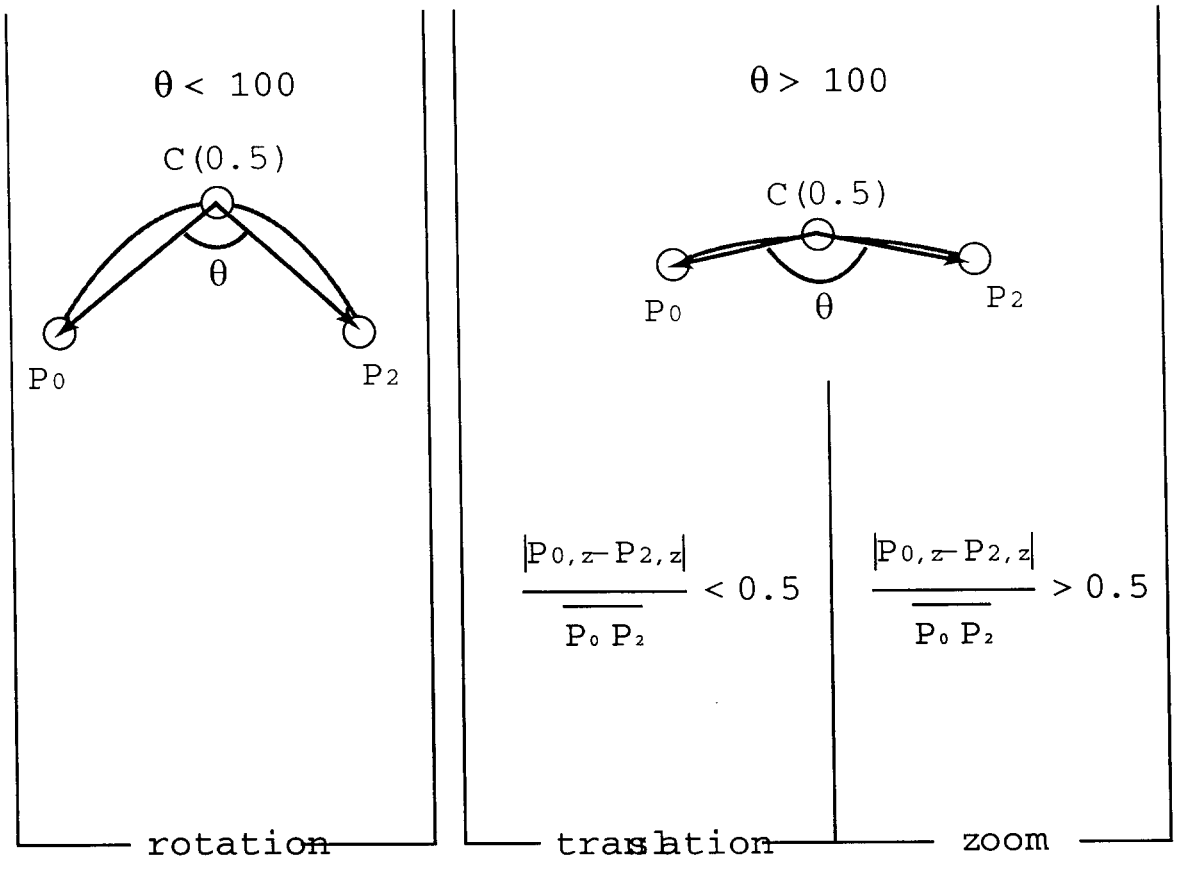


Figure 4: Determining gesture type using Bezier curve.

($C(0.5)$), we compute the inside angle of triangle inside the curve (θ) by

$$\vec{v_{10}} = C(0.5) - P_0 \tag{15}$$

$$\vec{v_{12}} = C(0.5) - P_2 \tag{16}$$

$$\vec{v_{10}} \cdot \vec{v_{12}} = |\vec{v_{10}}||\vec{v_{12}}|cos(\theta) \tag{17}$$

We classify "flat" trajectory as translation or zooming while "curved" as rotation. The inside angle would be smaller if the curve has high curvature. So if $\theta \leq 100°$, then we classify the gesture as rotation. Otherwise, the gesture is translation or zooming. If the movement was mostly in z direction, then we recognize it as "zooming" or otherwise "translation." To do so, we compute the ratio of the change of depth between first and last control points vs. distance between them.

$$\frac{|P_{2,z} - P_{0,z}|}{|P_2 - P_0|} \tag{18}$$

If the distance is greater than 0.5 then it is classified as zoom, otherwise it would be a translation. Since the method is geometry-based, the entire trajectory is recorded and can be used to recognize gestures by the hand speed.

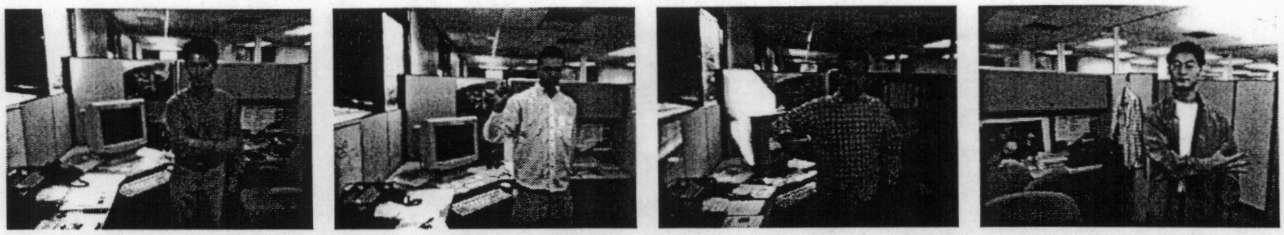


Figure 5: Subjects participated in the experiment. Note the variations in lighting condition, difference in skin tone, and a complex background.

3 Results

Experimental setup consists of a Digiclops system (Point Grey Research, [22]) on a Pentium 4 PC 1.5GHz with 512 MB RAM. The system is based on a triangulation between three cameras. Since the camera parameters (their relative positions, the focal length and resolution) are fixed, re-calibration is not usually required. The results are organized in four sections: manipulating hand detection; manipulation mode detection; gesture recognition; and overall performance. Testing data set includes 100 gestures, 226 manipulation modes, and 1641 frames (with a 320 $\times$ 240 image size) of manipulating hands of four people. We have presented initial manipulating hand tracking results on a smaller data set in [20]. The data was captured in an indoor setting with varying natural light and a complex background. Four different people with various skin tones participated in the experiments (refer to Figure 5).

3.1 Manipulating Hand Detection

The results of detecting manipulating hands (described in Section 2.2) are shown in Table 1. We detected the manipulating hand correctly in 1639 of 1641 images (99.9%). Two incidences of misdetection are shown in Figure 6. In both instances, detection failed because the depth readings of the manipulating hand and face were identical. In the upper image, the face was at [0.0323449, -0.35879, 1.72748] and the hand was at [-0.465777, -0.0511539, 1.72748]. For the lower image, the face was at [0.0323449, -0.35879, 1.72748] and the hand was at [-0.360104, 0.0452926, 1.72748]. In these two cases, the histogram-based range filtering was not effective enough to robustly compute the 3-D location.

Table 1: Manipulating hand detection results for ten sequences.

	correct	incorrect			
data set	manipulating hand	non-manipulating hand	face other body parts	non-skin regions	total manipulating hand
1	98	0	0	0	98
2	199	0	0	0	199
3	98	0	0	0	98
4	198	0	0	0	198
5	50	0	0	0	50
6	199	0	0	0	199
7	200	0	0	0	200
8	198	0	2	0	200
9	199	0	0	0	199
10	200	0	0	0	200
total	1639	0	2	0	1641
total (%)	99.9	0.0	0.1	0.0	100.0

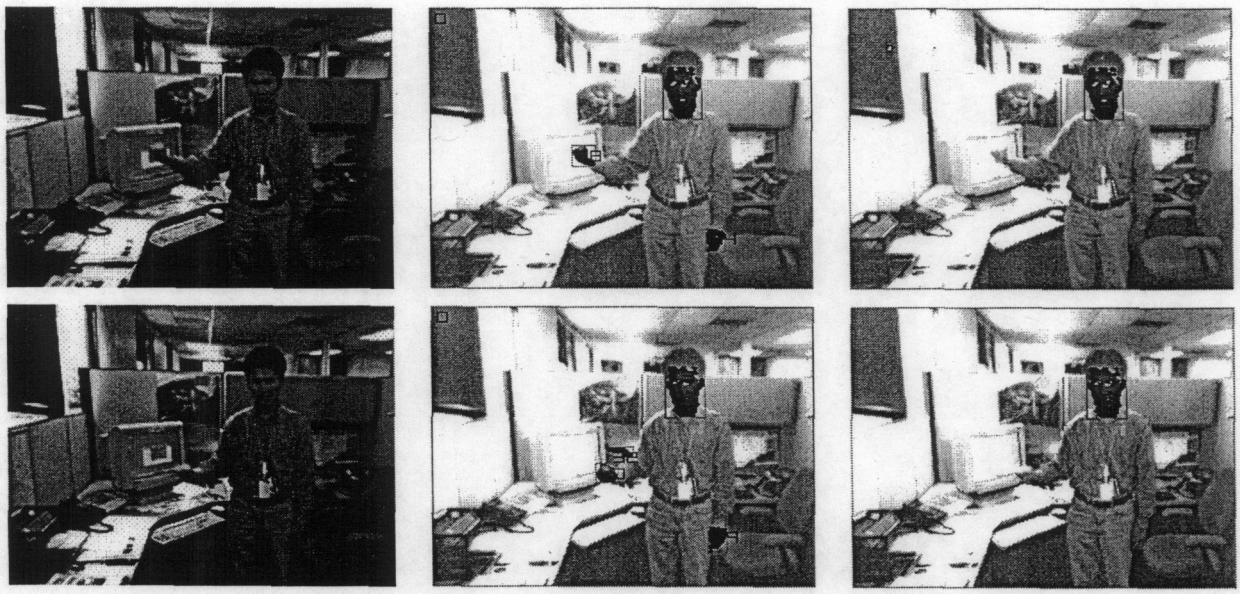


Figure 6: Two incidences of misdetection of manipulating hand. The color images (left), skin regions (mid), and detected manipulating hand (right) are shown.

3.2 Manipulation Detection

Evaluating Manipulation Detection. Each manipulation mode is defined by the starting frame, end frame, and its mode (on or off). We evaluated the manipulation mode detection in four categories: correct, incorrect, over, and under detection as shown in Figure 7. Detected manipulation mode on and off is shown as a bar as the frame number increase from left to right. Upper bar is for Algorithm Detected (AD) and lower bar is for Ground Truth (GT).

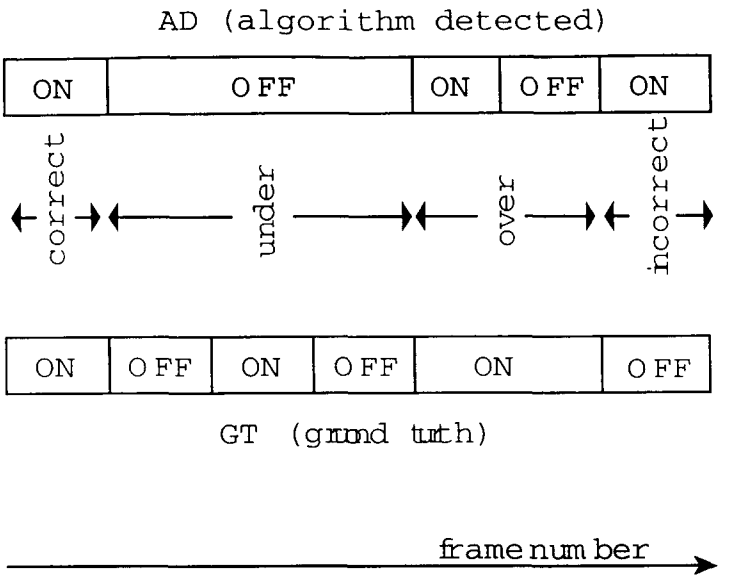


Figure 7: Evaluating Manipulation Mode Detection.

Four possible outcomes are classified below:

- **Correct detection.** AD and GT detections with at least 90% overlap in duration and same mode.
- **Incorrect detection.** AD and GT detections with at least 90% overlap in duration but different mode.
- **Over detection.** Multiple AD detections with at least 90% overlap in duration with one GT detection with same mode.
- **Under detection.** Multiple GT detections with at least 90% overlap in duration with one AD detection with same mode.

Results. The results of manipulation mode detections are shown in Table 2. The method correctly identified 216 of the 226 modes (95.6%). All five underdetections were due to missing the manipulation mode change (hand opening or closing). For these five missed detections, the area change was 63%, 74.7%, 71.4%, 68%, and -54%, which did not meet the threshold of +/-75%. The hand detections of 63% and -54% are shown in Figure 8.

Table 2: Manipulating Mode Detection

data set	correct	incorrect	over	under	total
1	19	0	0	0	19
2	35	0	0	0	35
3	11	0	0	1	13
4	24	0	0	1	25
5	8	0	0	0	8
6	30	0	0	0	30
7	31	0	0	0	31
8	16	0	0	1	19
9	18	0	0	1	21
10	24	0	0	1	25
total	216	0	0	5	226
total (%)	95.6	0.0	0.0	2.2	100.0

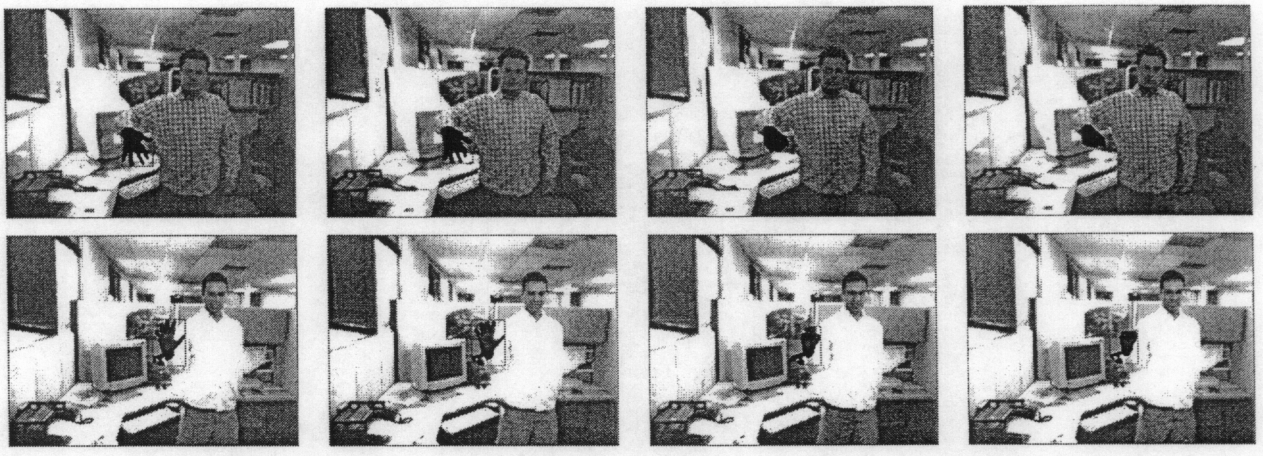


Figure 8: Two misidentifications of Manipulation Modes. Upper with area change of 63% and lower with area change of -54% do not meet the threshold of +/-75%.

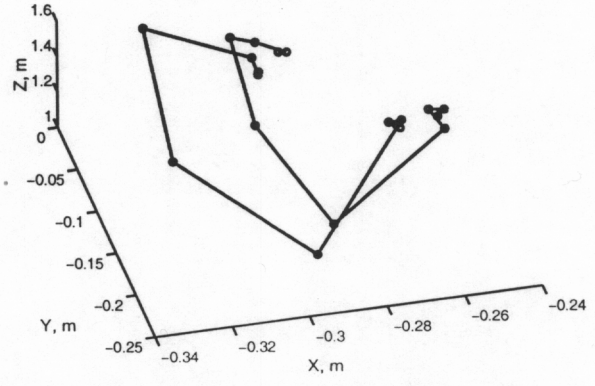


Figure 9: 3-D trajectories of two incorrectly recognized gestures.

Table 3: Gesture Recognition Performance

data set	correct	incorrect			total
		zoom	translation	rotation	
1	7	0	0	0	7
2	14	0	0	0	14
3	6	0	0	0	6
4	10	0	0	0	10
5	4	0	0	0	4
6	13	0	0	0	13
7	12	0	0	0	12
8	7	0	0	0	7
9	8	2	0	0	10
10	12	0	0	0	12
total	93	2	0	0	95
total (%)	97.9	2.1	0	0	100.0

3.3 Gesture Recognition

In order to compute the results of gesture recognition, we have taken the correctly identified manipulation modes and checked the recognition rate. Note that recognition rate is evaluating the goodness of recognition rate by using Bezier curve representation. The overall performance of evaluating the correctness of recognition versus intended recognition is shown in the next section. The results are shown in Table 3. The algorithm identified gestures correctly 93 out of 95 times (97.9%). For the two incorrect recognitions, the zoom gestures were recognized as rotation by estimating the inner angle (θ) as 104.9 and 125.124. The trajectory plots of two gestures are shown in Figure 9. The reason for the failure was due to using three-control points bezier curve, and the curves in question were too concave to be adequately described by three control points.

Motion trajectories for defined object manipulations in a 3-D space are shown in Figure 10. It is apparent that zoom gesture (blue) represents a significant change in Z coordinate relative to the almost constant depth during the translation (green), which is a predominantly X-axis motion. Rotational semicircle is also well-defined. Trajectory analysis is domain specific. Currently, we are working with HCI experts to determine necessary sets of gestures for controlling different visualization packages.

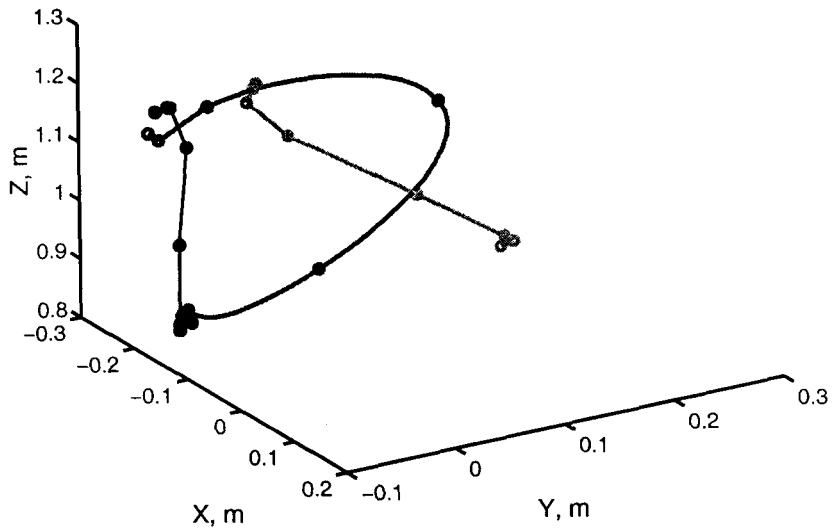


Figure 10: Motion trajectories in a 3-D space.

3.4 Overall Performance

Overall performance is evaluating the number of correctly recognized gestures vs. the number of intended gestures (shown in Table 4.) The algorithm correctly recognized 93 out of 100 intended gestures. Note that in the previous section, only 95 gestures were considered because 5 were not detected due to misdetection of manipulation mode as shown in section 3.2. Table 4 also shows the reasons for five non-correct classifications of gestures. We have divided those five incidences into "incorrect" if the gesture was incorrectly recognized as wrong gesture and "missed" if the gesture was not detected. "Incorrect" and "missed" are then subdivided into failures of three steps: "hand" (manipulating hand detection), "mode" (manipulating mode detection), and "recognition" (gesture recognition). Two gestures were incorrectly recognized due to a small angle as mentioned in Section 3.3. Three gestures were missed because the mode detection underdetected the gesture as mentioned in Section 2. The system executed at the rate of 5.5 frames/sec.

3.5 Application to Virtual Manipulation

Figure 11 shows the application of recognized gestural commands to a small virtual object. Complex virtual objects (3D views of two chromosomes) have been selected to provide a visual illustration of discussed operations. The dataset shown represents visualization of 3D reconstruction of fruit fly chromosomes based on 2D slices of CT scan images. The results of three sequentially applied virtual manipulations through

Table 4: Overall Recognition Performance.

data set	correct	incorrect			missed			total
		hand	mode	recognition	hand	mode	recognition	
1	7	0	0	0	0	0	0	7
2	14	0	0	0	0	0	0	14
3	6	0	0	0	0	1	0	7
4	10	0	0	0	0	1	0	11
5	4	0	0	0	0	0	0	4
6	13	0	0	0	0	0	0	13
7	12	0	0	0	0	0	0	12
8	7	0	0	0	0	1	0	8
9	8	0	0	2	0	1	0	11
10	12	0	0	0	0	1	0	12
total	93	0	0	2	0	5	0	100

gestures of rotation, zoom, and translation are shown in Figure 11. Both recognition and visualization are in real-time. The latter part is written in OpenGL. Real-time visualization of large data sets represents a significant bottleneck, which is currently under investigation by various research groups.

3.6 Environment (Two-Handed) Operations

So far we discussed only basic visualization commands. However, the technique described is easily expandable to larger sets of gestures covering more advanced data manipulations. For example, environment operations are applicable when several objects have to be affected. Such operations include: 1) (environment) stretch,

2) (environment) squeeze,

3) (environment) bend,

4) (environment) translation,

5) (environment) rotation, and

6) (environment) zoom.

All of these manipulations require two-handed gestures. When two simultaneous "ON" switches of the manipulation mode occur close in the depth, this signals that an environment command is being processed. Gestures (4)-(6) are defined similarly to their respective one-handed equivalents. They are distinguished from (1)-(3) by having an approximately constant distance between hands (Figure 12). Gesture (3) is characterized by the angle between trajectories, while (1) and (2) occur along a straight line and differ

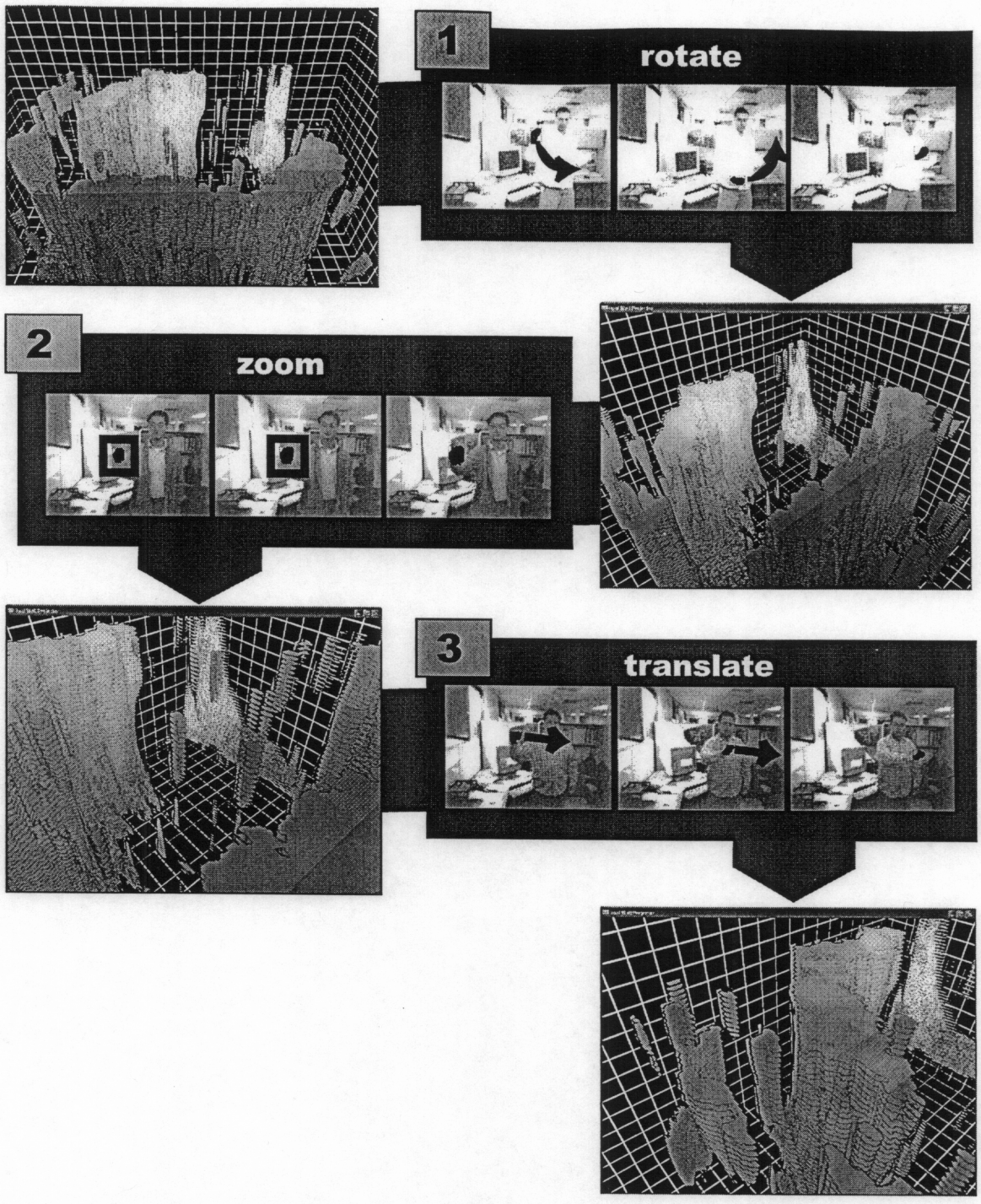


Figure 11: Example of Virtual Manipulation Application. Three sequential manipulations are shown. Red arrows are superimposed on images to enhance the appearance of gesture trajectories.

only in direction. Appearance of a virtual environment after a horizontal stretch operation is shown in Figure 13.

Figure 12: Frame samples for two-handed gestures (top to bottom, respectively): rotation, translation, zoom, bend, stretch and squeeze.

4 Conclusions

Visual data exploration has tremendous capabilities for revealing properties and abnormalities in large data sets. This paper described a gesture recognition system for visualization navigation. Scientists are

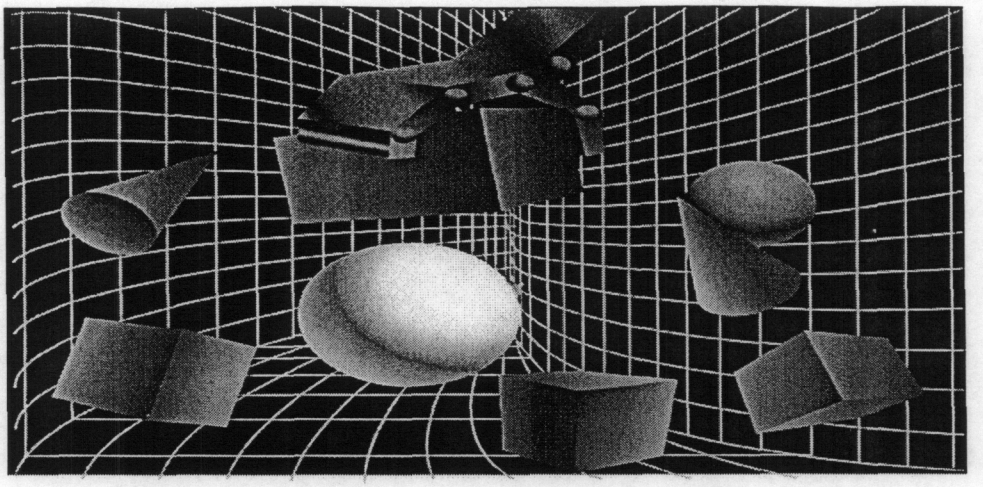


Figure 13: Appearance of a virtual environment after a horizontal stretch operation.

interested in developing interactive settings for exploring large data sets in an intuitive environment. The input consists of registered 3-D data. Bezier curves are used for trajectory analysis and classification of gestures. A geometric (rather than statistical) model is motivated by the nature of the application (representation of visualization operations) and its requirements (robustness and simplicity). Since the method is geometry-based, the entire trajectory is recorded and can be used to recognize gestures by the hand speed. Variations in the hand speed within a gesture as well as between gestures are normalized by parameterizing the points along the trajectory to a Bezier curve description with respect to the hand speed at each frame. The method is robust and reliable: correct hand identification rate is 99.9% (from 1639 frames), modes of hand movements are correct 95.6% of the time, recognition rate (given the right mode) is 97.9%. An application to gesture-controlled visualization of 3D bioinformatics data is also presented. Future work includes defining a larger set of gestures based on the basic movements already analyzed, using gesture speed for classification, and creating actual hand models for detection of smaller, more subtle gestures.

Acknowledgments

We would like to thank the LLNL VIEWS Visualization project for Figure 1(a); the example data set appears as Figure 1(b) courtesy of Art Mirin of LLNL.

References

[1] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. on PAMI*, 19(7):780–785, July 1997.

[2] M.-H. Yang and N. Ahuja. Recognizing hand gestures using motion trajectories. In *Proceedings of IEEE CS Conference on Computer Vision and Pattern Recognition*, volume 1, pages 466–472, Fort Collins, CO, June 1999.

[3] J. M. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: an application to human hand tracking. *Proc. of European Conference on Computer Vision*, 2:35–46, May 1994.

[4] J. J. Kuch and T.S. Huang. Model-based tracking of self-occluding articulated objects. In *Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration*, pages 666–671, Cambridge, MA, June 1995.

[5] R. Cutler and L. Davis. Real-time periodic motion detection, analysis, and applications. In *Proceedings of IEEE CS Conference on Computer Vision and Pattern Recognition*, volume 2, pages 326–332, Fort Collins, CO, June 1999.

[6] D. J. Moore, I. A. Essa, and M. H. Hayes III. Exploiting human actions and object context for recognition tasks. In *Proceedings of International Conference on Computer Vision*, pages 80–86, Corfu, Greece, September 1999.

[7] Y. Iwai, H. Shimizu, and M. Yachida. Real-time context-based gesture recognition using hmm and automaton. In *Proceedings of International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 127–134, Corfu, Greece, September 1999.

[8] C. Vogler, H. Sun, and D. Metaxas. A framework for motion recognition with applications to American sign language and gait recognition. In *Proc. Workshop on Human Motion*, pages 33 – 38, Los Alamitos, CA, December 2001.

[9] Y. Yacoob and M. J. Black. Parameterized modeling and recognition of activities. *Journal of Computer Vision and Image Understanding*, 73(2):232–247, 1999.

[10] A. F. Bobick and A. D. Wilson. A state-based approach to the representation and recognition of gesture. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(12):1325–1337, December 1997.

[11] R. Grzeszcuk, G. Bradski, M. H. Chu, and J.-Y. Bouguet. Stereo based gesture recognition invariant to 3D pose and lighting. In *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, volume 1, pages 826–833, Hilton Head Island, SC, June 2000.

[12] P. Hong, M. Turk, and T. S. Huang. Constructing finite state machines for fast gesture recognition. In *Proc. 15th International Conference on Pattern Recognition*, volume 3, pages 691–694, Barcelona, Spain, September 2000.

[13] Y. Sato, M. Saito, and H. Koike. Real-time input of 3D pose and gestures of a user's hand and its applications for HCI. In *Proc. IEEE Virtual Reality*, pages 79–86, Yokohama, Japan, March 2001.

[14] H. Hongo, M. Ohya, M. Yasumoto, and K. Yamamoto. Face and hand gesture recognition for human-computer interaction. In *Proc. 15th International Conference on Pattern Recognition*, volume 2, pages 921 – 924, Barcelona, Spain, September 2000.

[15] H. S. Yoon, J. Soh, Y. J. Bae, and H. S. Yang. Hand gesture recognition using combined features of location, angle and velocity. *Pattern Recognition*, 34(7):1491–1501, July 2001.

[16] L. Gupta and S. Ma. Gesture-based interaction and communication: Automated classification of hand gesture contours. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 31(1):114–120, February 2001.

[17] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. In *IEEE Nonrigid and Articulated Motion Workshop*, pages 90–102, San Juan, Puerto Rico, June 1997.

[18] V. I. Pavlovic, R. Sharma, and T. S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Trans. on PAMI*, 19(7):677–695, July 1997.

[19] M. W. Powell and R. Murphy. Position estimation of micro-rovers using a spherical coordinate transform color segmenter. In *Proceedings of IEEE Workshop on Photometric Modeling for Computer Vision and Graphics*, pages 21–27, Fort Collins, CO, June 1999.

[20] L. V. Tsap and M. C. Shin. Improving quality and speed of range data for communication. In *Proc. IEEE International Workshop on Cues in Communication.*

[21] L. Piegl and W. Tiller. *The NURBS Book.* Springer-Verlag, New York, 1997.

[22] Point Grey Research Inc. *Digiclops Stereo Vision System, User's guide and command reference.* Inc., Point Grey Research, Vancouver, BC, 2000.

A Appendix Least Squares Fitting of 3-D Bezier Curves

With three control points, we have only three unknowns: x, y, and z in P_1 (the middle control point). First, the nth-degree Bezier curve is described as

$$C(u) = \sum_{i=0}^{n} B_{i,n}(u) P_i \tag{19}$$

With three control points,

$$N_{0,2} = B_{0,2} = (1-u)^2 \tag{20}$$

$$N_{1,2} = B_{1,2} = 2u(1-u) \tag{21}$$

$$N_{2,2} = B_{2,2} = u^2 \tag{22}$$

N is now $(m-1) \times 1$ ($(m-1) \times (n-1)$ and $n = 2$) matrix and

$$N^T N = \sum_{k=1}^{m-1} N_{1,p}(\bar{u}_k) \qquad = \sum_{k=1}^{m-1} 2\bar{u}_k(1-\bar{u}_k) \tag{23}$$

We set P to be

$$P = \begin{bmatrix} P_{1,x} & P_{1,y} & P_{1,z} \end{bmatrix} \tag{24}$$

R_k is now simplified and R is a 1x3 matrix.

$$R_k = Q_k - (1-\bar{u}_k)^2 Q_0 - \bar{u}_k^2 Q_m \tag{25}$$

$$R = \begin{bmatrix} R_x & R_y & R_z \end{bmatrix}^T \tag{26}$$

$$= \left[\sum_{k=1}^{m-1} 2\bar{u}_k(1-\bar{u}_k) R_k \right] \tag{27}$$

The middle control point is the solution to $(N^T N)P = R$ which is defined as

$$P_{1,x} = \frac{R_x}{(N^T N)} \tag{28}$$

$$P1_{1,y} = \frac{R_y}{(N^T N)} \tag{29}$$

$$P1_{1,z} = \frac{R_z}{(N^T N)} \tag{30}$$